Macros

This chapter covers the following topics:

- General Information
- Defining a Macro
- Calling a Macro
- Local Macro Variables

General Information

A macro is a collection of Con-form instructions and/or text which is useful when the same sequence of instructions or the same passage of text is required repeatedly within one or several source documents.

A macro can contain any Con-form instructions, with the exception of a second .MA instruction (i.e. macros cannot be nested). A macro can contain further macro calls.

You must not define recursive macros (i.e. a macro must not call itself, neither directly nor through other macros).

As a rule, all macros you define should be contained in a separate document, the so-called "formatting profile" which is always processed before your source document (see *Formatting Profiles*).

When you do not define a macro in the formatting profile, you can only use it in the document in which it has been defined. In this case, the macro call is only recognized as a valid instruction when it occurs after the macro definition.

Another possibility to make a macro generally available in your cabinet is to create a separate document for each macro definition. You must then specify the following instructions in your source text to call the macro:

```
.EM documentname
.macroname
```

The *documentname* in the above example is the name of the document that contains the macro definition (see the .EM instruction). The *macroname* is the name of the macro that is to be called.

Defining a Macro

.MA - Start Macro

```
.MA macro-name
```

The .MA instruction is used to indicate the beginning of the macro definition.

The .MA instruction must be followed by the name of the macro in the same line. The name can be up to 100 characters long. Upper-case and lower-case letters in the macro name are not distinguished.

For example, to define a macro called "defaults", you must specify:

```
.MA defaults
```

Tip:
It is recommended that a macro name has at least three characters, so that it cannot be confused with a Con-form instruction.

You can now specify the macro definition (i.e. all required instructions and/or text lines) below the .MA instruction. The following is an example of a macro which contains your default settings:

```
.MA defaults
.LM 10;.RM 60
.FI ON;.JU ON
.OP ESC=/
.OP HYP=E
.IF &$LC < 5:.NP
.ME
```

You must specify the .ME instruction after the macro definition to define the end of the macro.

Disabling a Standard Con-form Instruction

When you use the name of a standard Con-form instruction as the name of a macro, the standard instruction is no longer processed. Instead of the standard instruction, the macro with the same name is processed every time it is specified in the source document.

This feature can be used to disable a standard Con-form instruction. For example, the macro below has no effect, since no instruction or text has been defined for it. However, it disables the Con-form instruction .NP which normally causes a form feed. This is useful when you want to print a document for proofreading where it is more important to save paper than to maintain the final page layout.

```
.MA NP
.ME
```

.MX - Exit from Macro

```
.MX
```

You can insert the .MX instruction at any point within the macro definition to exit the macro before its end is reached and to continue processing after the .ME instruction. The macro definitions after the .MX instruction are not processed.

When you use the .MX instruction with the .IF instruction, it is recommended that you only use it with a simple consequence that does not require the .EI instruction.

Note:
When you use the .MX instruction with a compound consequence (.TH) or alternative (.EL) and you exit from the macro before the .EI instruction occurs, Con-form does not recognize that you have specified the .EI instruction and outputs an error message.

.ME - End Macro

```
.ME
```

The .ME instruction is used to indicate the end of the macro definition.

You can optionally specify the name of the macro after the .ME instruction. For example:

```
.ME defaults
```

It is not possible to conditionally suppress the processing of the .ME instruction by using it in an .IF construction.

Calling a Macro

To call a macro, you must specify its name in the source text, preceded by a period. For example, to call the macro with the name "defaults", you must specify:

```
.defaults
```

As a result, the instructions and/or text lines which have been defined for the macro are processed. Processing begins immediately following the .MA instruction and continues until either an .MX instruction or the .ME instruction which indicates the end of the macro is encountered.

Local Macro Variables

When you define local macro variables, you can pass, for example, a name and address as parameters to a macro which produces a standard letter.

In addition to the local macro variables, you can also use variables which have been defined with the .SV instruction or system variables in the macro definition or as parameters in the macro call. However, you should be careful when you design macros which alter the values of these variables; confusion can easily result if several macros modify the same variable.

Substitution (see the .SU instruction) is automatically switched on by a macro call *with* parameters. Substitution is not switched on by a macro call without parameters.

Defining a Macro that Requires Parameters

You can include up to 99 local macro variables in the macro definition. The local macro variables are named $1, $2, $3, ... $99. Each local macro variable in the macro definition must be preceded by the variable character. Initially, the variable character is the ampersand (&).

Later, when you call the macro, you must specify the parameters (i.e. the appropriate values for the variables) after the macro name (see *Calling a Macro that Requires Parameters*).

In the following macro definition, the variable $1 determines the number of blank lines to be inserted in the document text, and variable $2 determines the text to be output below the blank lines:

```
.MA figure
.IL &$1
.CE 1
&$2
.SL 2
.ME
```

The variable $1 in the macro definition must be the first parameter in the macro call. The variable $2 in the macro definition must be the second parameter in the macro call. Thus, to call the above macro, you must specify the macro call with two parameters:

```
.figure 5 Macros
```

As a result, 5 blank lines (variable $1) are inserted in the document text and the centered text string "Macros" (variable $2) is output below the blank lines.

Determining the Number of Parameters in the Macro Call

The local macro variable $0 contains the number of parameters which have been specified after the macro name. For example:

```
.figure 5 Macros
Number of parameters in the current macro call: &$0
```

In the above example, two parameters have been defined after the macro name. As a result, the value 2 is assigned to the local macro variable $0. Thus, above example causes the following formatted output:

```
Number of parameters in the current macro call: 2
```

When you call a macro without a parameter, the value 0 (zero) is assigned to the local macro variable $0.

Ensuring that the Macro is Called with the Correct Number of Parameters

You can use the local macro variable $0 to ensure that a macro is processed with the correct number of parameters. For example:

```
.MA figure
.IF &$0 NE 2
.TH
*******************************************************
This macro requires two parameters.
Parameter 1: Number of lines to be inserted for the figure.
Parameter 2: The caption for the figure.
*******************************************************
.EL
```

```
.IL &$1
.CE 1
&$2
.EI
.ME
```

In the above example, the .IF instruction is used to ensure that exactly two parameters are specified with the macro call. When more or fewer than 2 parameters are defined, a message is output.

Calling a Macro that Requires Parameters

When a macro requires parameters, you must supply the appropriate values after the macro name - separated from it by a single space. For example:

```
.chap1 Introduction
```

Enclosing a Parameter Within Apostrophes

If a parameter contains spaces or commas, you must enclose the parameter within apostrophes. For example:

```
.chap1 'How to Use this Manual'
```

When you omit the apostrophes in the above example, Con-form tries to process five parameters, each word being a separate parameter.

Note:
Macro parameters that do *not* include spaces or commas may be enclosed within apostrophes if desired, but this is not necessary.

You can include an apostrophe in a macro parameter which is enclosed within apostrophes by repeating the apostrophe. For example:

```
.chap1 'All in a Day''s Work'
```

You can include apostrophes within a parameter which is not enclosed within apostrophes. For example:

```
.chap1  Day's
```

Any number of apostrophes may appear within the parameter as long as the first character is not an apostrophe. For example:

```
.chap1 All'in'a'Day's'Work
```

Defining More Than One Parameter

If the macro has more than one parameter, you must insert a comma and/or space between two parameters. The example below has two parameters, separated by a comma (it is also possible to separate the parameters by a space):

```
.example 15, 'Defining a Macro'
```

Each parameter you specify in the macro call is assigned to a local macro variable. In the above example, the value 15 is assigned to the variable $1 and the text string "Defining a Macro" is assigned to the variable $2.

The local macro variables are only available within the called macro. As soon as the execution of the macro is completed, the local macro variables are no longer available.